From an architectural point of view, there are three main advantages of the swarm-robotics approach (Martinoli, 2001). First, the resulting collective systems are scalable, because the control architecture is exactly the same from a few to thousands of robots. Second, such systems are flexible, because individual robots can be dynamically added or removed without explicit reorganization by the operator. Third, these systems are robust, not only due to robot redundancy but also due to the minimalist robot design.

The approaches derived or inspired by social insects demonstrated to be feasible and interesting, though they are not the single alternative for the control and coordination of groups of small robots. The reader may be disappointed by the great simplicity of the tasks reported here as the state of the art in swarm robotics. Besides, collective robotics, including swarm-robotics, still lacks a rigorous theoretical foundation. In spite of these apparent drawbacks, it must be kept in mind that swarm robotics may still be one of the most useful strategies for collective robotics in the light of miniaturization; that is, micro- and nano-robots, though much more has to be done than has actually already been accomplished.

5.4. SOCIAL ADAPTATION OF KNOWLEDGE

The two previously described sections focused on how the social behavior of insect societies led to the development of computational algorithms for problem solving and strategies for the coordination of collective robotics. The *particle swarm* (PS) algorithm, to be discussed in this section, has as one of its motivations to create a simulation of human social behavior; that is, the ability of human societies to process knowledge (Kennedy and Eberhart, 1995; Kennedy, 1997; Kennedy, 2004). As is characteristic of all swarm intelligence approaches (actually, most biologically inspired algorithms), PS also takes into account a population of individuals capable of interacting with the environment and one another, in particular some of its neighbors. Thus, population level behaviors will emerge from individual interactions. Although the original approach has also been inspired by particle systems (Chapter 7) and the collective behavior of some animal societies, the main focus of the algorithm is on its social adaptation of knowledge; the same focus taken here.

A very simple sociocognitive theory underlies the particle swarm approach (Kennedy et al., 2001; Kennedy, 2004). The authors theorize that the process of cultural adaptation comprises a low-level component corresponding to the actual behavior of individuals, and a high-level component in the formation of patterns across individuals. Therefore, each individual within a population has its own experience and they know how good it is and, as social beings, they also have some knowledge of how other neighboring individuals have performed.

These two types of information correspond to *individual learning* and *cultural or social transmission*, respectively. The probability that a given individual takes a certain decision is a function of how successful this decision was to him/her in the past. The decision is also affected by social influences, though the exact rules in human societies are not very clear (Kennedy, 2004; Kennedy et

al., 2001). In the particle swarm proposal, individuals tend to be influenced by the best successes of anyone they are connected to, i.e., the members of their social (sociometric) neighborhood that have had the most success so far.

Kennedy (1998) and Kennedy et al. (2001) use three principles to summarize the process of cultural adaptation. These principles may be combined so as to enable individuals to adapt to complex environmental challenges:

- *Evaluate*: the capability of individuals to sense the environment allows them to quantify their degree of *goodness* (quality or suitability) in relation to some parameter(s) or task(s), such as height when one wants to become a basketball player.
- *Compare*: people usually use others as standards for assessing themselves, what may serve as a kind of motivation to learn and change. For instance, we look at other people and evaluate our wealth, look, humor, beauty, exam mark, whom to vote, and so forth.
- *Imitate*: human imitation comprises taking the perspective of someone else, not only by imitating a behavior but by realizing its purpose and executing the behavior of others when it is appropriate. Imitation is central to human sociality and the acquisition and maintenance of mental abilities.

5.4.1. Particle Swarm

In the particle swarm (PS) algorithm, individuals searching for solutions to a given problem learn from their own past experience and from the experiences of others. Individuals evaluate themselves, compare to their neighbors and imitate only those neighbors who are superior to themselves. Therefore, individuals are able to evaluate, compare and imitate a number of possible situations the environment offers them.

Although there are two basic versions of the PS algorithm, binary and real-valued, this section focuses only on the latter one, namely, the real-valued or continuous PS. This is mainly because this version has a much broader applicability and it has been more extensively studied as well. Under this perspective, the PS algorithm can be viewed as a numeric optimization procedure inspired by the social adaptation of knowledge discussed above. It searches for optima in an L-dimensional real-valued space, $\Re^L$. Therefore, the discussion regarding problem solving as a search in a search space presented in Section 3.2 is also appropriate for the purposes of this section.

In real-valued spaces, the variables of a function to be optimized can be conceptualized as a vector that corresponds to a point in a multidimensional search space. Multiple individuals can thus be plotted within a single set of coordinates, where a number of individuals will correspond to a set of points or particles in the space. Individuals similar to one another in relevant features appear closer to one another in the space, as illustrated in Figure 5.15.

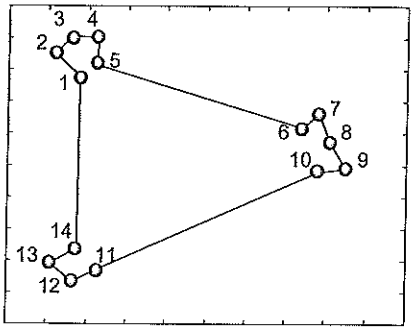


Figure 5.15: Particles in a bi-dimensional real-valued search-space. Particles are connected to their topological neighbors, and neighbors tend to cluster in the same regions of the space.

The individuals are viewed as points in a search-space and their change over time is represented as movements. In this case, individuals truly correspond to *particles*. Forgetting and learning are seen as a decrease or increase on some dimensions, attitude changes are seen as movements between the negative and positive ends of an axis, and emotion and mood changes of numerous individuals can be plotted conceptually in a coordinate system. The swarm of particles corresponds to the multiple individuals in the population.

One insight from social psychology is that these particles will tend to move toward one another and to influence one another as individuals seek agreement with their neighbors (Kennedy et al., 2001; Kennedy, 2004). Another insight is that the space in which the particles move is heterogeneous with respect to evaluation; that is, some regions are better than others, meaning that they have a higher goodness value.

In mathematical terms, the PS algorithm can be implemented as follows. The position of a particle i is given by $\mathbf{x}_i$, which is an L-dimensional vector in $\Re^L$. The change of position of a particle is denoted by $\Delta\mathbf{x}_i$, which is a vector that is added to the position coordinates in order to move the particle from one iteration t to the other $t+1$:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \Delta\mathbf{x}_i(t+1) \tag{5.21}$$

The vector $\Delta\mathbf{x}_i$ is commonly referred to as the velocity $\mathbf{v}_i$ of the particle. The particle swarm algorithm samples the search-space by modifying the velocity of each particle. The question that remains is how to define the velocity of the particle so that it moves in the appropriate directions and with the appropriate "step size" in the search-space. In addition, the neighborhood of each particle (individual) has to be defined.

The inspiration taken from the social-psychological sciences suggests that individuals (particles) should be influenced by their own previous experience and

the experience of its neighbors. Neighborhood here does not necessarily mean similarity in the parameter space; instead, it refers to topological similarity in a given structure of the population. In a social network, a given individual particle is capable of influencing the ones to which it is socially connected. Neighborhood is thus defined for each individual particle based on its position in a topological array, usually implemented as a ring structure (the last member is a neighbor of the first one).

There are a number of different schemes to connect the individuals of the population. Most particle swarm implementations use one of two simple sociometric principles. The first, called *gbest* (*g* for global), conceptually connects all members of the population to one another (Figure 5.16(a)). The effect of this is that each particle is influenced by the very best performance of any member of the entire population. The second, termed *lbest* (*l* for local), creates a neighborhood for each individual comprising itself and its k-nearest neighbors in the population (Figure 5.16(a)). To illustrate, consider the particles in Figure 5.15: each individual particle is connected with its 2-nearest neighbors in an *lbest* scheme ($k = 2$). Thus, particle $\mathbf{x}_3$ has neighbors $\mathbf{x}_2$ and $\mathbf{x}_4$, particle $\mathbf{x}_8$ has neighbors $\mathbf{x}_7$ and $\mathbf{x}_9$, and so on.

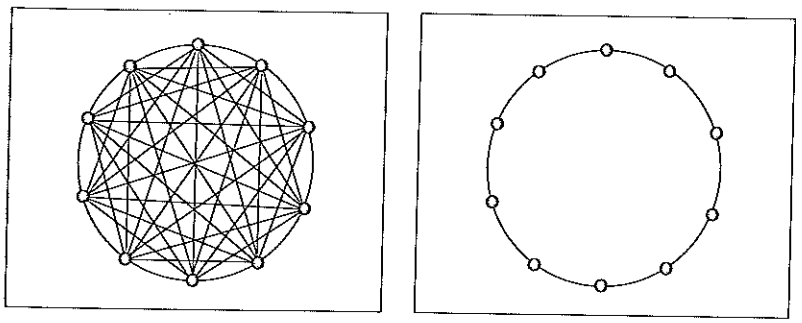
A particle will move in a certain direction as a function of its current position $\mathbf{x}_i(t)$, its change in position $\Delta\mathbf{x}_i(t)$, the location of the particle's best success so far $\mathbf{p}_i$, and the best position found by any member of its neighborhood $\mathbf{p}_g$:

$$\mathbf{x}_i(t+1) = f(\mathbf{x}_i(t), \Delta\mathbf{x}_i(t), \mathbf{p}_i, \mathbf{p}_g) \tag{5.22}$$

The influence of the terms $\Delta\mathbf{x}_i(t)$, $\mathbf{p}_i$, and $\mathbf{p}_g$ can be summarized by a change $\Delta\mathbf{x}_i(t+1)$ to be applied at iteration $t+1$:

$$\Delta\mathbf{x}_i(t+1) = \Delta\mathbf{x}_i(t) + \varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i(t)) + \varphi_2 \otimes (\mathbf{p}_g - \mathbf{x}_i(t)) \tag{5.23}$$

where φ_1 and φ_2 represent positive random vectors composed of numbers drawn from uniform distributions with a predefined upper limit: $\varphi_1 = U(0, AC_1)$ and $\varphi_2 = U(0, AC_2)$; $U(0, AC)$ is a vector composed of uniformly distributed random numbers, and AC is called the *acceleration constant*.



(a) (b)

Figure 5.16: Illustration of *gbest* and *lbest* neighborhoods. (a) *gbest*. (b) *lbest* for $k = 2$.

According to Kennedy (2004), the sum of the two acceleration constants should equal 4.1, $AC_1 + AC_2 = 4.1$ (usually both are 2.05). The second term on the right hand side of Equation (5.23) is proportional to the difference between the particle's previous best and its current position, and the last term on the right hand side of Equation (5.23) is proportional to the difference between the neighborhood's best and the current position of the particle. The symbol $\otimes$ represents the elementwise vector multiplication.

In (Kennedy, 1997), the author clearly identifies the meaning of each term in Equation (5.23) and performs a number of experiments, with a single application problem, in order to evaluate the importance of "social" and "cognitive" interactions in the PS algorithm. He defines Equation (5.23) without the last term as the "cognition-only" model, and without the second term, but with all the others, as the "social-only" model.

In order to limit the change in position of a particle so that the system does not "explode", two values v_{min} and v_{max} are defined for the change Δx, thus guaranteeing that the particles oscillate within some predefined boundaries:

If $\Delta x_{id} > v_{max}$, then $\Delta x_{id} = v_{max}$,

Else if $\Delta x_{id} < v_{min}$ then $\Delta x_{id} = v_{min}$

```
procedure [X] = PS(max_it, AC1, AC2, vmax, vmin)
    initialize X   //usually xi, ∀i, is initialized at random
    initialize Δxi //at random, Δxi ∈ [vmin, vmax]
    t ← 1
    while t < max_it do,
       for i = 1 to N do,          //for each particle
            if g(xi) > g(pi),
                then pi = xi,      //best indiv. performance
            end if
            g = i                  //arbitrary
            //for all neighbors
            for j = indexes of neighbors
                if g(pj) > g(pg),
                    then g = j,    //index of best neighbor
                end if
            end for
            Δxi ← Δxi + φ1⊗(pi − xi) + φ2⊗(pg − xi)
            Δxi ∈ [vmin, vmax]
            xi ← xi + Δxi
       end for
       t ← t + 1
    end while
end procedure
```

Algorithm 5.4: Standard particle swarm optimization (PS) algorithm.

Let $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ be the swarm of particles, $g(\mathbf{x}_i)$ the goodness of particle $\mathbf{x}_i$, $\Delta\mathbf{x}_i$ its change in position, $v_{\min}$ and $v_{\max}$ the lower and upper limit for the change in position (velocity), respectively, and L the dimension of the particles. Assume a maximal number of iterations `max_it` to be run and a swarm of a given size N (according to the authors, the swarm size is usually $N \in [10,50]$). The original PS algorithm is described in Algorithm 5.4. In the section on applications (Section 5.4.2), two simple modifications of the algorithm that considerably improve performance will be introduced.

Note that solving a problem using the PS algorithm involves the same three aspects suggested in the previous chapter: the choice of a representation (in this case it is fixed: real-valued vectors), the choice of an objective function, and the choice of a goodness function according to the desired objective.

5.4.2. Selected Applications from the Literature: A Brief Description

Algorithm 5.4 presents the original particle swarm optimization algorithm developed taking inspiration from the social adaptation of knowledge and bird flocking behavior. The description presented here emphasized the social behavior of human beings as the primary motivation for the algorithm because this is the emphasis given in Kennedy et al. (2001) and Kennedy (2004), two of the most important fundamental works of the field.

To illustrate in what types of problem the algorithm has been applied to, two applications will be presented. The first example - optimization of neural network weights - illustrates the importance of the PS algorithm as a tool to be hybridized with other strategies. The second example - numerical function optimization - presents two slight variations of the original algorithm and discusses its potentiality to solve multi-dimensional function optimization problems in general.

Optimization of Neural Network Weights

One of the first applications of the particle swarm algorithm was to the problem of defining appropriate weights for artificial neural networks (Kennedy and Eberhart, 1995). It has been discussed in the previous chapter that a neural network is comprised of a structured set of nodes partially or fully connected. The neural network performs a mapping from a set of input data into one or more output node(s). Each network node is a processing unit that receives stimuli from the environment or (an)other node(s), processes these inputs and produces an output signal. Assuming that the nodes in the network are homogeneous; that is, of same (predefined) type, and the network structure is of a fixed given size, the resultant network learning task is reduced to the problem of determining an appropriate weight set that will lead to a satisfactory network behavior when given input stimuli are presented. Figure 5.17 illustrates a simple network with two input nodes, one output node, and two intermediary nodes between the input and output nodes.

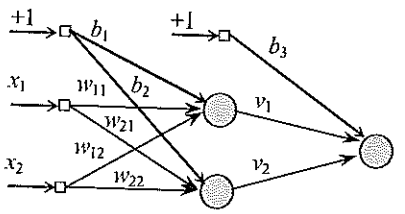


Figure 5.17: An example of a simple artificial neural network with two input nodes, two intermediate or hidden nodes, and one output node.

Inputs		Output
x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

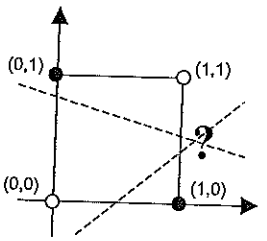


Figure 5.18: The XOR function and its graphical representation. The black dot corresponds to 1, while the circles are equivalent to 0.

Each connection between the nodes has an assigned weight w_{ij}, v_i, and b_i, where w and v correspond to weights of different layers in the network, and b_i corresponds to the bias of a node. Usually, the weight vectors of this type of network are determined according to a learning rule or algorithm, responsible for updating the network weights along an iterative procedure of adaptation. In the present example, a PS algorithm will be used to determine the network weights instead of a standard learning algorithm.

Consider the problem of using the neural network of Figure 5.17 to determine a mapping capable of simulating the exclusive-or (XOR) logic function depicted in Figure 5.18. The difficulty in solving this problem, seen as a classification problem, resides in the fact that it is not possible to separate the input patterns that lead to an output 1 from the input patterns that result in an output 0 by simply drawing a line or a plane on the graph. That means that the two classes available (0 or 1) are not linearly separable. This is a classic benchmark problem for the network structure presented in Figure 5.17, though nowadays there are various algorithms that solve this problem very easily.

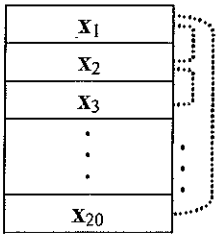


Figure 5.19: The structured neighborhood used in the PS implementation to define neural network weights. Particle $\mathbf{x}_1$ is connected to $\mathbf{x}_2$ and $\mathbf{x}_{20}$, particle $\mathbf{x}_2$ is connected to $\mathbf{x}_1$ and $\mathbf{x}_3$, and so forth. This is a circular neighborhood such as the one depicted in Figure 5.16(b).

In the experimental results reported in (Kennedy and Eberhart, 1995; Kennedy, 1997; Kennedy et al., 2001), the authors used a swarm composed of 20 particles ($N = 20$) initialized at random. Each particle has a dimension $L = 9$ corresponding to the nine weights of the network ($w_{11}, w_{12}, w_{21}, w_{22}, b_1, b_2, b_3, v_1, v_2$) to be determined. The swarm of particles can be defined by using a matrix $\mathbf{X}$ with 20 rows and 9 columns, $\mathbf{X} \in \Re^{20\times 9}$, where each row corresponds to a particle of dimension $L = 9$.

The neighborhood is of the 2-nearest neighbor type, as illustrated in Figure 5.19. The goodness of each particle is defined as the error (e.g., the mean squared error, MSE) between the current network output and the desired outputs given in Figure 5.18. The goal is thus to determine an appropriate weight set that minimizes the mean squared error between the network output for all input patterns and the desired output.

As each particle corresponds to one of the parameters to be adjusted in the neural network, the movements of the particles in the search space allow for a selection of particles that provide the desired network performance. In their simulations the desired criterion was $MSE < 0.02$. Experimental results were very satisfactory, and presented for different values of v_{max}.

Numerical Function Optimization

Similarly to the application of evolutionary algorithms, Angeline (1998), and Shi and Eberhart (1999) applied the PS algorithm to perform numeric function optimization. The authors used some standard test functions (e.g., Sphere, Rosenbrock, Rastrigin, Griewank) taken from the literature of evolutionary algorithms to assess the performance of the PS algorithm:

Sphere $$f_0(x) = \sum_{i=1}^{L} x_i^2$$

Rosenbrock $$f_1(x) = \sum_{i=1}^{L} \left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right)$$

Rastrigin $$f_2(x) = \sum_{i=1}^{L} \left(x_i^2 - 10.\cos(2\pi x_i) + 10\right)$$

Griewank $$f_3(x) = \frac{1}{4000}\sum_{i=1}^{L} x_i^2 - \prod_{i=1}^{L} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

In (Shi and Eberhart, 1999), all functions were implemented using $L = 10$, 20 and 30 dimensions, and the PS algorithm was run for 1000, 1500, and 2000 iterations, respectively. Also, the authors studied different population sizes for the PS, $N = 20$, 40, 80 and 160. In this application, each particle corresponds to one candidate solution to the problem, in a form similar to that used in genetic algorithms, but with real-valued particles instead of a binary representation. They suggested that by using a decreasing factor termed *inertia weight* w the PS algorithm was able to better balance exploration with exploitation:

$$\Delta\mathbf{x}_i(t+1) = w.\,\Delta\mathbf{x}_i(t) + \varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i(t)) + \varphi_2 \otimes (\mathbf{p}_g - \mathbf{x}_i(t)) \qquad \textbf{(5.24)}$$

where φ_1 and φ_2 were defined as previously and w was assigned an initial value but was allowed to be reduced (or cooled, such as the temperature in simulated annealing) during the iterative procedure of adaptation. The decreasing inertia weight started at 0.9 and was linearly decreased until $w = 0.4$. The limiting values for $\Delta\mathbf{x}$ were set equal to v_{max} according to a pre-defined table.

Another variant of the original algorithm that is currently known as the standard particle swarm algorithm (Kennedy, 2004) involves the use of a global constriction coefficient χ (Clerc and Kennedy, 2002) as follows:

$$\Delta\mathbf{x}_i(t+1) = \chi\,(\Delta\mathbf{x}_i(t) + \varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i(t)) + \varphi_2 \otimes (\mathbf{p}_g - \mathbf{x}_i(t))) \qquad \textbf{(5.25)}$$

where the parameter $\chi \approx 0.729$.

There are some java applets that show the movement of particles when applied to some of these functions. Most of them use the decreasing inertia weight. Some of the codes and demos available about the PSO algorithm can be found at the PSO website: http://www.swarmintelligence.org.

5.4.3. Scope of Particle Swarm Optimization

The original applications of PS focused primarily on neural network parameter adjustment and model selection. This corresponds to the problem of designing neural networks, including the network weights adjustment and structure definition processes. Model selection has been one of the most important application areas of evolutionary algorithms as well. Actually, some similarities can be observed between these approaches, but their distinctions are also undeniable; for instance, there can be no evolution if there is no selection, and PS does not account for selection, individuals move but never die. Kennedy et al. (2001) and Kennedy (2004) provide a good discussion about the similarities and differences between PS and other approaches.

PS algorithms have been applied to a number of numeric and parameter optimization problems, including real-world tasks. For instance, works can be found in the literature applying the PS approach to tasks like human tremor analysis, milling optimization, ingredient mix optimization, reactive power and voltage control, battery pack state-of-charge estimation, and improvised music composition.

5.4.4. From Social Systems to Particle Swarm

Section 5.4 discussed the importance of the social influence for the acquisition and improvement of knowledge. Some of these concepts led to the proposal of the particle swarm algorithm. Table 5.3 summarizes how to interpret the social-psychological views presented as a particle swarm approach.

Table 5.3: Interpretation of the sociocognitive domain into the particle swarm approach.

Social-Psychology	PS Algorithms
Individual (minds)	Particles in space
Population of individuals	Swarm of particles
Forgetting and learning	Increase or decrease in some attribute values of the particle
Individual own experience	Each particle has some knowledge of how it performed in the past and uses it to determine where it is going to move to
Social interactions	Each particle also has some knowledge of how other particles around itself performed and uses it to determine where it is going to move to

5.4.5. Summary of Particle Swarm Optimization

Although not explicitly discussed in the chapter that introduces the PS algorithm in Kennedy's book (Kennedy et al., 2001; Chapter 7), the algorithm was also motivated by the social behavior of organisms such as bird flocking and fish schooling (Kennedy and Eberhart, 1995). The algorithm is not only a tool for optimization, but also a tool for representing sociocognition of human and artificial agents, based on principles of social psychology. Some scientists suggest that knowledge is optimized by social interaction and thinking is not only private but also a result of interactions with other people. There are reports in the literature of feral humans, human children who grew up in the wild are commonly uneducable, unable to learn or to speak, unable to adapt to social life, and never able to show much evidence of cognition or thinking (Kennedy et al., 2001; Kennedy, 2004).

The PS algorithm as an optimization tool provides a population-based search procedure in which individuals, called particles, change their position with time. In real-valued PS, the particles fly around in a multidimensional search-space. During flight, each particle adjusts its position according to its own experience and according to the experience of a number of neighboring particles, making use of its best position so far and the best position of its neighbors as well. Therefore, the PS algorithm combines local search with global search, attempting to balance exploration with exploitation.

A particle swarm is thus another self-organizing system whose global dynamics emerge from local interactions. As each individual trajectory is adjusted toward its success and the success of neighbors, the swarm converges or clusters in optimal regions of the search-space.

5.5. SUMMARY

This chapter has discussed four main approaches in swarm intelligence: ant colony optimization, ant clustering, swarm robotics, and particle swarm. Ant colony optimization algorithms are suitable for finding minimal cost paths on a graph, mainly in those cases in which other more classical algorithms cannot be efficiently applied. Although each ant of the colony builds a solution to the problem at hand, good quality solutions can only emerge from the cooperative behavior of the colony. The communication among ants is performed indirectly via pheromone trails. Note also that the ants themselves are not adaptive individuals. Instead, they adapt the environment by releasing pheromone, thus, changing the way the problem is represented and perceived by other ants. This is one example of a stigmergic algorithm.

The clustering of dead bodies and larval sorting in ants has led to the development of the ant clustering algorithm. ACA is suitable for exploratory data analysis, in particular for the clustering of unlabelled data sets. It works by projecting the data into a bi-dimensional grid in which ants are allowed to move about carrying items to and from specific cells. Regions containing a large number of similar data reinforce the release of similar data, and vice-versa.

Most collective robotic systems were inspired by the collective behavior of ant colonies. Foraging, clustering of dead bodies, clustering around food, and collective prey retrieval have motivated the development of robotic systems to perform equivalent tasks. These systems are believed to be much promising when miniaturization is pursued. In such cases, they can aid in the unmanned exploration of inaccessible environments, in the large scale manufacturing of products, and in the medical sciences and surgery, among other things. The swarm robotics systems described here are other examples of stigmergic systems.

The particle swarm approach is inspired by the human social adaptation of knowledge and has demonstrated good performances in a number of applications, such as numeric function optimization and optimal design of other natural computing techniques (e.g., neural networks). The algorithm is conceptually

simple and easily implementable. Although this chapter has only presented its most well-known version, real-valued PS, binary and discrete versions of the PS algorithm are also available.

5.6. EXERCISES

5.6.1. Questions

1. Describe how the ant colony of the movie "Antz" is organized. Contrast the perspective of ant colonies presented in the movie with that of real ant colonies.
2. How many queens an ant or a termite colony may have? Discuss.
3. It is known that scientists do not have a detailed knowledge of the inner workings of insect swarms. The identification of the rules that govern individual behaviors is a major challenge, and without such knowledge it is hard to develop appropriate models and thus to predict their behavior.

   Can this (apparent) unpredictability of behavior be a drawback of the swarm intelligence approaches inspired by social insects?

   Would it be possible that a swarm of robots goes out of control due to this lack of knowledge?
4. The Ant Farm project of the MIT Artificial Intelligence Lab had as one of its final goals to design a structured community of micro-robots inspired by the social behavior of ants. By running the video recording of some of their experiments (http://www.ai.mit.edu/projects/ants/social-behavior.html), it is possible to observe a recruitment behavior of robots. Based upon the discussion presented in Section 5.3.3 and the information available at their website, identify which type of communication is being performed by the robots: direct or indirect communication. Comment on its efficiency.

   Can this recruitment strategy be used in conjunction with the work presented in Kube's videos (Section 5.3.3)? Discuss.
5. Section 5.3 was dedicated to swarm robotics. Suggest two real-world applications (different from those presented here) for each of these systems and discuss how these could be accomplished by swarm robotics.
6. In Section 2.2.3, it was described the basic behavior of termites' nest building. It was seen that, during nest building, termites are stimulated to work according to the current configuration of the local environment and other termites. In this case, the accumulation of pellets reinforces the dropping of more pellets in a given portion of the space. The intermediate results of their building process are pillars, while the final result is a mound. Is the model presented in Section 5.2.4 of the present chapter for the clustering of dead bodies and larval sorting appropriate to model termite nest building as well? Discuss.